



National Textile University

Department of Computer Science

Subject:

Operating System

Submitted to:

Sir Nasir Mehmood

Submitted by:

Eman Babar

Reg. number:

23-NTU-CS-FL-1148

Semester:

5th- A

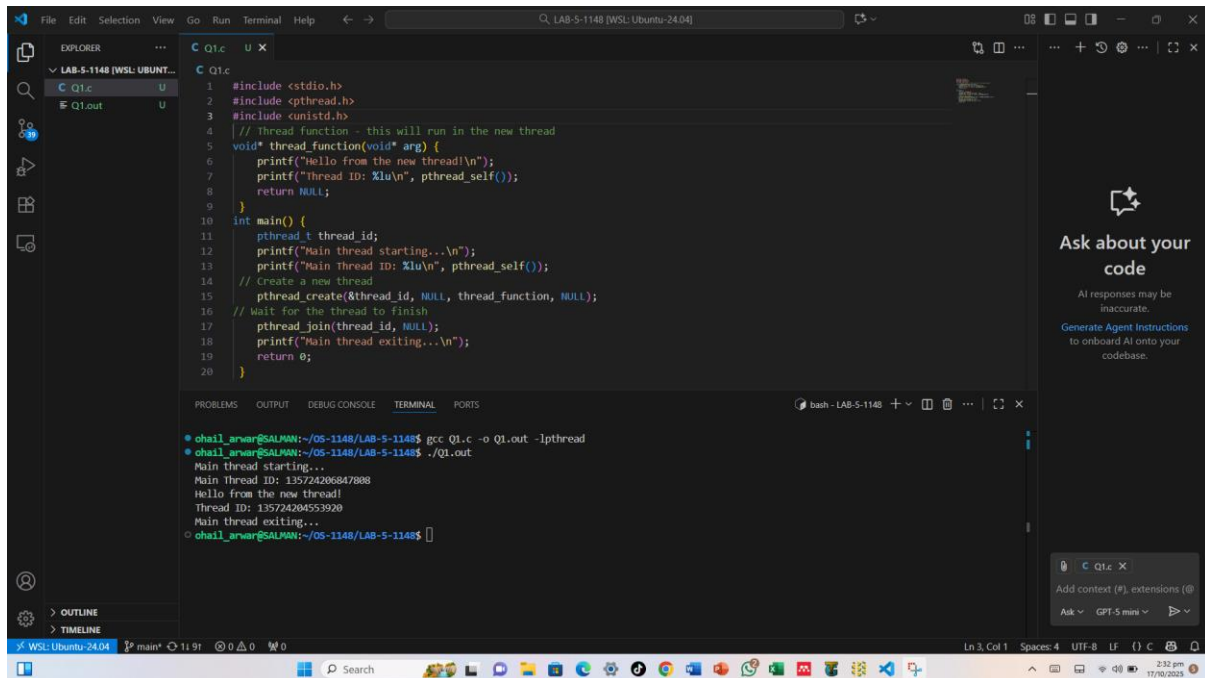
LAB-05: Introduction to Threads

Program 1: Creating a Simple Thread

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
// Thread function - this will run in the new thread
void* thread_function(void* arg) {
    printf("Hello from the new thread!\n");
    printf("Thread ID: %lu\n", pthread_self());
    return NULL;
}
int main() {
    pthread_t thread_id;
    printf("Main thread starting...\n");
    printf("Main Thread ID: %lu\n", pthread_self());
    // Create a new thread
    pthread_create(&thread_id, NULL, thread_function, NULL);
    // Wait for the thread to finish
    pthread_join(thread_id, NULL);
    printf("Main thread exiting...\n");
    return 0;
}
```

Output:

The screenshot shows the Visual Studio Code interface with a C file named 'Q1.c' open. The code is the same as provided in the previous block. The terminal at the bottom shows the execution of the program. The output is as follows:

```
o chail_arwar@SALMAN:~/05-1148/LAB-5-1148$ gcc Q1.c -o Q1.out -lpthread
o chail_arwar@SALMAN:~/05-1148/LAB-5-1148$ ./Q1.out
Main thread starting...
Main Thread ID: 135724206847808
Hello from the new thread!
Thread ID: 135724204553920
Main thread exiting...
```

Program 2: Passing Arguments to Threads

Code:

```
#include <stdio.h>
#include <pthread.h>
void* print_number(void* arg) {
```

```
// We know that we've passed an integer pointer
int num = *(int*)arg; // Cast void* back to int*
printf("Thread received number: %d\n", num);
printf("Square: %d\n", num * num);
return NULL;
}

int main() {
    pthread_t thread_id;
    int number = 42;
    printf("Creating thread with argument: %d\n", number);
    // Pass address of 'number' to thread
    pthread_create(&thread_id, NULL, print_number, &number);
    pthread_join(thread_id, NULL);
    printf("Main thread done.\n");
    return 0;
}
```

Output:

```
chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ gcc Q2.c -o Q2.out -lpthread
chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ ./Q2.out
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$
```

Program 3: Doubling the CGPA

Code:

```
#include <stdio.h>
#include <pthread.h>
void* print_number(void* arg) {
    // We know that we've passed an integer pointer
    float num = *(float*)arg; // Cast void* back to int*
    printf("Thread received number: %f\n", num);
    printf("Double: %f\n", num * 2);
    return NULL;
}

int main() {
    pthread_t thread_id;
    float number = 3.2;
    printf("Creating thread with argument: %f\n", number);
    // Pass address of 'number' to thread
```

```

pthread_create(&thread_id, NULL, print_number, &number);
pthread_join(thread_id, NULL);
printf("Main thread done.\n");
return 0;
}

```

Output:

The screenshot shows a Visual Studio Code editor with a C program in a file named `Q3.c`. The program defines a function `print_number` that takes a `void*` argument, casts it to a `float`, and prints it. The `main` function creates a thread with the `print_number` function, passing the value `3.2`. The terminal output shows the program's execution, including the thread's output and the main thread's completion message.

```

1 #include <stdio.h>
2 #include <pthread.h>
3 void* print_number(void* arg) {
4     // We know that we've passed an integer pointer
5     float num = *(float*)arg; // Cast void* back to int*
6     printf("Thread received number: %f\n", num);
7     printf("Double: %f\n", num * 2);
8     return NULL;
9 }
10
11 int main() {
12     pthread_t thread_id;
13     float number = 3.2;
14     printf("Creating thread with argument: %f\n", number);
15     // Pass address of 'number' to thread
16     pthread_create(&thread_id, NULL, print_number, &number);
17     pthread_join(thread_id, NULL);
18     printf("Main thread done.\n");
19     return 0;
20 }

```

```

chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ gcc Q3.c -o Q3.out -lpthread
chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$ ./Q3.out
Creating thread with argument: 3.200000
Thread received number: 3.200000
Double: 6.400000
Main thread done.
chail_arwar@SALMAN:~/OS-1148/LAB-5-1148$

```

Program 4: Passing Multiple Data

Code:

```

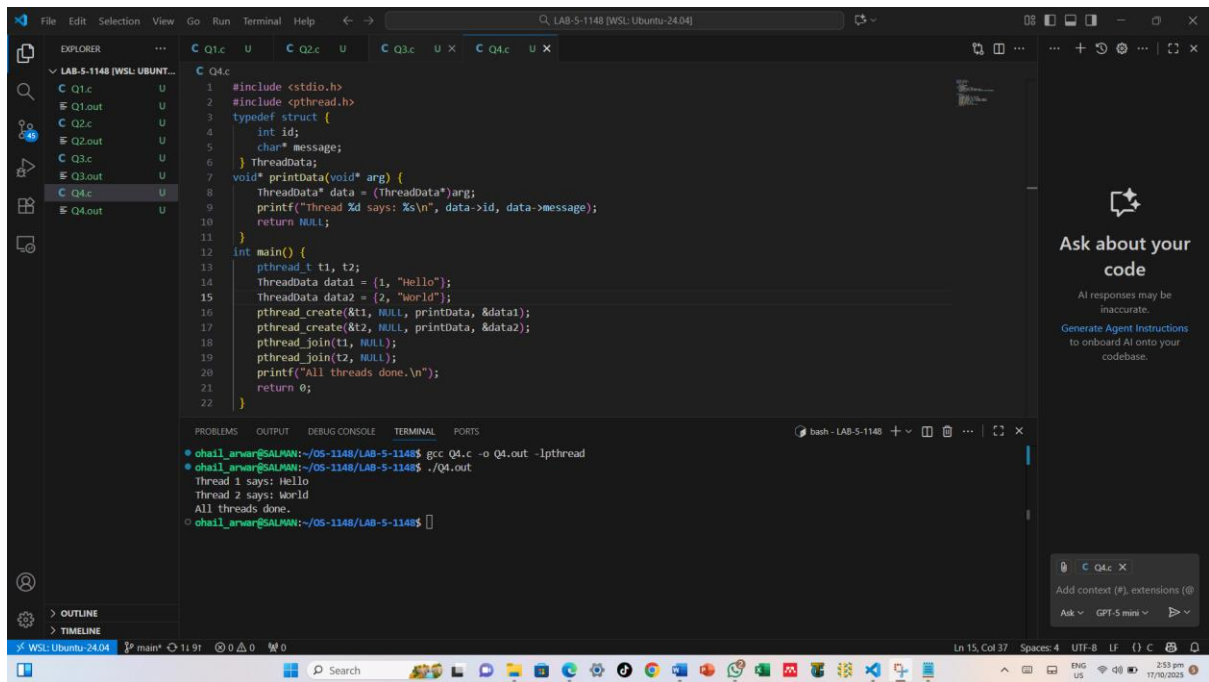
#include <stdio.h>
#include <pthread.h>
typedef struct {
    int id;
    char* message;
} ThreadData;

void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %d says: %s\n", data->id, data->message);
    return NULL;
}

int main() {
    pthread_t t1, t2;
    ThreadData data1 = {1, "Hello"};
    ThreadData data2 = {2, "World"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("All threads done.\n");
    return 0;
}

```

Output:



Program 5: Showing Name and CGPA

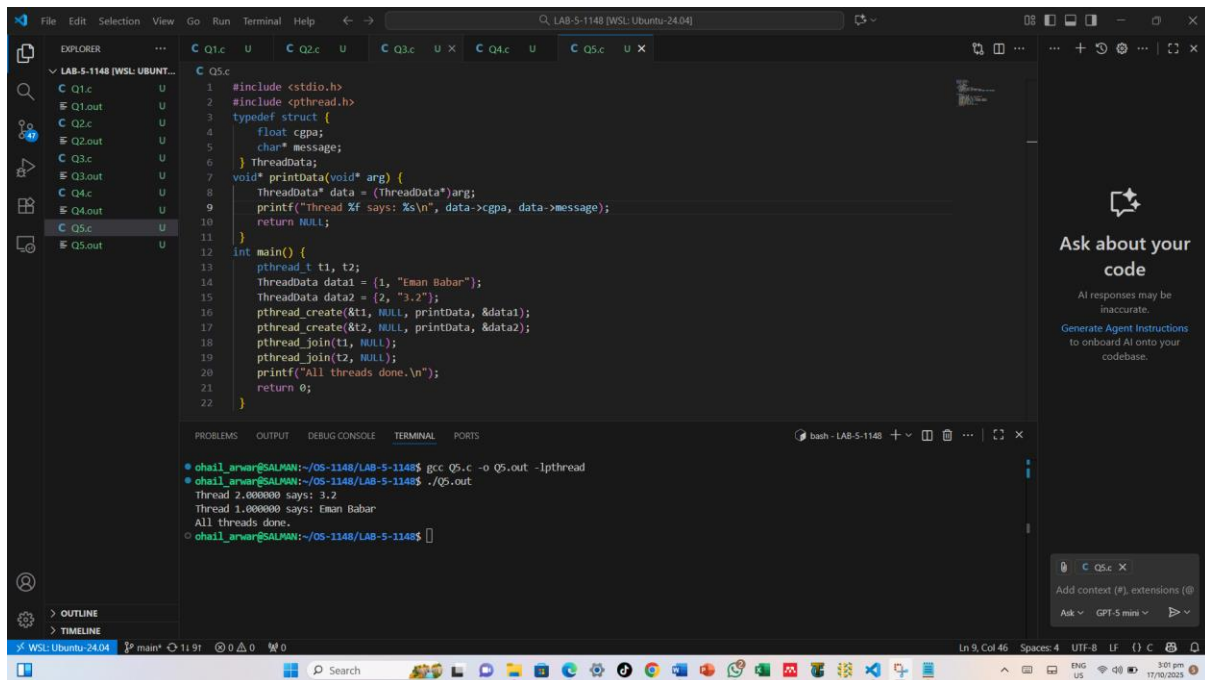
Code:

```

#include <stdio.h>
#include <pthread.h>
typedef struct {
    float cgpa;
    char* message;
} ThreadData;
void* printData(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    printf("Thread %f says: %s\n", data->cgpa, data->message);
    return NULL;
}
int main() {
    pthread_t t1, t2;
    ThreadData data1 = {1, "Eman Babar"};
    ThreadData data2 = {2, "3.2"};
    pthread_create(&t1, NULL, printData, &data1);
    pthread_create(&t2, NULL, printData, &data2);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("All threads done.\n");
    return 0;
}

```

Output:



Program 6: Thread Return Values

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
void* calculate_sum(void* arg) {
    int n = *(int*)arg;
    int* result = malloc(sizeof(int)); // Allocate memory for result
    *result = 0;
    for (int i = 1; i <= n; i++) {
        *result += i;
    }
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
    return (void*)result; // Return the result
}
int main() {
    pthread_t thread_id;
    int n = 100;
    void* sum;
    pthread_create(&thread_id, NULL, calculate_sum, &n);
    // Get the return value from thread
    pthread_join(thread_id, &sum);
    printf("Main received result: %d\n", *(int*)sum);
    free(sum); // Don't forget to free allocated memory
    return 0;
}
```

Output:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>

void* calculate_sum(void* arg) {
    int n = *(int*)arg;
    int* result = malloc(sizeof(int)); // Allocate memory for result
    *result = 0;
    for (int i = 1; i <= n; i++) {
        *result += i;
    }
    printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
    return (void*)result; // Return the result
}

int main() {
    pthread_t thread_id;
    int n = 100;
    void* sum;
    pthread_create(&thread_id, NULL, calculate_sum, &n);
    // Get the return value from thread
    pthread_join(thread_id, &sum);
    printf("Main received result: %d\n", *(int*)sum);
    free(sum); // Don't forget to free allocated memory
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
o chail_arwar@SALMAN:~/05-1148/LAB-5-1148$ gcc Q6.c -o Q6.out -lpthread
o chail_arwar@SALMAN:~/05-1148/LAB-5-1148$ ./Q6.out
thread calculated sum of 1 to 100 = 5050
Main received result: 5050
o chail_arwar@SALMAN:~/05-1148/LAB-5-1148$
```

Program 7: Thread Return Values

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void* worker(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting task...\n", thread_num);
    sleep(1); // Simulate some work
    printf("Thread %d: Task completed!\n", thread_num);
    return NULL;
}

int main() {
    pthread_t threads[3];
    int thread_ids[3];
    for (int i = 0; i < 3; i++) {
        thread_ids[i] = i + 1;
        pthread_create(&threads[i], NULL, worker, &thread_ids[i]);
    }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Main thread: All threads have finished.\n");
    return 0;
}
```

Output:


```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void* worker(void* arg) {
    int thread_num = *(int*)arg;
    printf("Thread %d: Starting task...\n", thread_num);
    sleep(1); // Simulate some work
    printf("Thread %d: Task completed!\n", thread_num);
    return NULL;
}

int main() {
    pthread_t threads[3];
    int thread_ids[3];
    for (int i = 0; i < 3; i++) {
        thread_ids[i] = i + 1;
        pthread_create(&threads[i], NULL, worker, &thread_ids[i]);
    }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Main thread: All threads have finished.\n");
    return 0;
}
```

Terminal Output:

```
ohai1_arwar@SALMAN:~/OS-1148/LAB-5-1148$ gcc Q7.c -o Q7.out -lpthread
ohai1_arwar@SALMAN:~/OS-1148/LAB-5-1148$ ./Q7.out
Thread 2: Starting task...
Thread 1: Starting task...
Thread 3: Starting task...
Thread 2: Task completed!
Thread 3: Task completed!
Thread 1: Task completed!
Main thread: All threads have finished.
```

Program 8: Demonstrating a Race Condition

Code:

```
#include <stdio.h>
#include <pthread.h>
int counter = 0; // Shared variable
void* increment(void* arg) {
    for (int i = 0; i < 100000; i++) {
        counter++; // Not thread-safe
    }
    return NULL;
}
int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("Expected counter value: 200000\n");
    printf("Actual counter value:  %d\n", counter);
    return 0;
}
```

Output:

The image shows a Visual Studio Code editor window with a C program that demonstrates a race condition using pthreads. The program is named `Q8.c` and is located in the `LAB-5-1148` directory. The code includes `stdio.h` and `pthread.h`, and defines a shared variable `counter`. Two threads, `t1` and `t2`, are created, each incrementing the counter 100,000 times. The main function prints the expected counter value (200,000) and the actual counter value (128,168), demonstrating a race condition.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 int counter = 0; // Shared variable
4 void* increment(void* arg) {
5     for (int i = 0; i < 100000; i++) {
6         counter++; // Not thread-safe
7     }
8     return NULL;
9 }
10 int main() {
11     pthread_t t1, t2;
12     pthread_create(&t1, NULL, increment, NULL);
13     pthread_create(&t2, NULL, increment, NULL);
14     pthread_join(t1, NULL);
15     pthread_join(t2, NULL);
16     printf("Expected counter value: 200000\n");
17     printf("Actual counter value: %d\n", counter);
18     return 0;
19 }
```

The terminal output shows the execution of the program:

```
o chail_arvar@SALMAN:~/05-1148/LAB-5-1148$ gcc Q8.c -o Q8.out -lpthread
o chail_arvar@SALMAN:~/05-1148/LAB-5-1148$ ./Q8.out
Expected counter value: 200000
Actual counter value: 128168
o chail_arvar@SALMAN:~/05-1148/LAB-5-1148$
```